iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.43353

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

TradeMarker: Artificial Intelligence Based Trademarks Similarity Search Engine

Yuvraj Singh[1], Sambhaji Mane[2], Vaibhav Dhumal[3]
[1, 2, 3] *Department of Computer Engineering, Parvatibai Genba Moze College of Engineering (WAGHOLI), SPPU, India*

Abstract: ***A trademark is a mark used by a company or a private human for the purpose of marking products or services that they manufacture or trade in. A restriction on the use of the trademark is necessary to enable sellers and manufacturers to build a reputation for themselves, to differentiate themselves from their competitors and thereby promote their businesses. In addition, the restriction also serves consumers and prevents their misuse by a name similar to another product. This restriction is done through the formal examination and approval of the trademarks. This process entails trademark examination against other approved trademarks which is currently a long manual process performed by experienced examiners. Current state-of-the-art trademark similarity search systems attempt to provide a single metric to quantify trademark similarities to a given mark [6, 7, 8, 9, 10, 11]. In this work we introduce a new way to carry out this process, by simultaneously conducting several independent searches on different similarity aspects - Automated content similarity, Image/pixel similarity, Text similarity, and Manual content similarity. This separation enables us to benefit from the advantages of each aspect, as opposed to combining them into one similarity aspect and diminishing the significance of each one of them***
Keywords: CNN, OCR, Image processing, Numpy, Scipy, Spyder

I. INTRODUCTION

A. Overview

A Trademark is a mark used in relation to goods for the purpose of indicating a connection between the goods and some person having the right as proprietor to use the mark.

1) *Motivation:* The motivation for these lawsuits could be based on more than just the noticeable similarity of the products under mention; they are also motivated by concepts of brand protection.
2) *Objective:* The project goal is to find similarity of text or brand logo. It is useful to uniquely identify or differentiate products. To easily find unique identification of products. To protect the good will that the trademark owner has built up in his products and services.

II. PROBLEM STATEMENT

A. Problem statement

A Trademark or trade mark is a distinctive sign or indication of some kind which is used by an individual, business or other legal entity to uniquely identify the source of its products and services to consumers to the most of the time brand name or logo if similar that time consumer not understand brand or product difference this problem overcomes to our application.

III. PROJECT REQUIREMENTS

A. External Interface Requirement

1) *User Interface:* Artificial Intelligence Based Trademarks Similarity Search Engine.
2) *Hardware Interfaces:* RAM: 8 GB as we are using Machine Learning Algorithm and Various High Level Libraries Laptop RAM minimum required is 8 GB. Hard Disk: 500 GB Data Set of CT scan images is to be used hence minimum 40 GB Hard Disk memory is required. Processor: Intel i5 Processor.
3) *Software Interfaces:* Operating System: Windows 10 IDE: Spyder Programming Language: Python

B. Non-Functional Requirement

1) *Performance Requirements:* The performance of the functions and every module must be well. The overall performance of the software will enable the users to work efficiently. Performance of encryption of data should be fast. Performance of the providing virtual environment should be a fast Safety Requirement. The application is designed in modules where errors can be detected and easily. This makes it easier to install and update new functionality if required.

2) *Safety Requirement:* The application is designed in modules where errors can be detected and fixed easily. This makes it easier to install and update new functionality if required.
3) *Software Quality Attributes:* Our software has many qualities attribute that are given below:
a) *Adaptability:* This software is adaptable by all users.
b) *Availability:* This software is freely available to all users. The availability of the software is easy for everyone.
c) *Maintainability:* After the deployment of the project if any error occurs then it can be easily maintained by the software developer.
d) *Reliability:* The performance of the software is better which will increase the reliability of the Software.
e) *User Friendliness:* Since, the software is a GUI application; the output generated is much user friendly in its behaviour.
f) *Integrity:* Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.
g) *Security:* Users are authenticated using many security phases so reliable security is provided. Testability: The software will be tested considering all the aspects.

IV. SYSTEM ANALYSIS

A. System Architecture

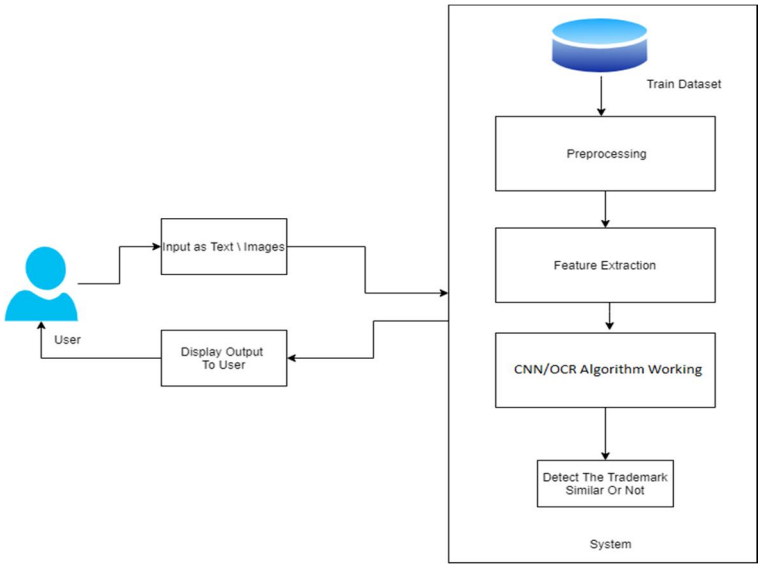


Fig. 1 System Architecture

1) *Module*
a) *Admin:* In this module, the admin has to log in by using a valid user name and password. After login successful he can do some operations, such as View All Users and Authorize, View All E-Commerce Website and Authorize, View All Products and Reviews, View All Products Early Reviews, View All Keyword Search Details, View All Products Search Ratio, View All Keyword Search Results, View All Product Review Rank Results.
b) *View and Authorize Users:* In this module, the admin can view the list of users who all registered. In this, the admin can view the user's details such as, user name, email, address and admin authorize the users.
c) *View Charts Results:* View All Products Search Ratio, View All Keyword Search Results, View All Product Review Rank Results.
d) *Ecommerce User:* In this module, there are n numbers of users present. Users should register before doing any operations. Once a user registers, their details will be stored in the database. After registration is successful, he has to login by using authorized user name and password Once Login is successful user will do some operations like Add Products, View All Products with reviews, View All Early Product's reviews, View All Purchased Transactions.
e) *End User:* In this module, there are n numbers of users present. Users should register before doing any operations. Once a user registers, their details will best or to the database. After registration is successful, he has to login by using authorized user name and password. Once Login is successful users will do some operations like Manage Account, Search Products by keyword and Purchase, View Your Search Transactions, View.

2) *Data Flow Diagram:* In Data Flow Diagram, we Show that flow of data in our system in DFD0 we show that base DFD in which rectangle present input as well as output and circle show our system, In DFD1 we show actual input and actual output of system input of our system is text or image and output is rumour detected likewise in DFD 2 we present operation of user as well as admin.

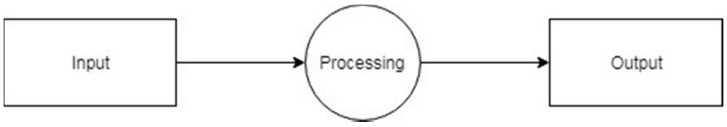


Fig. 2 Data Flow (0) diagram

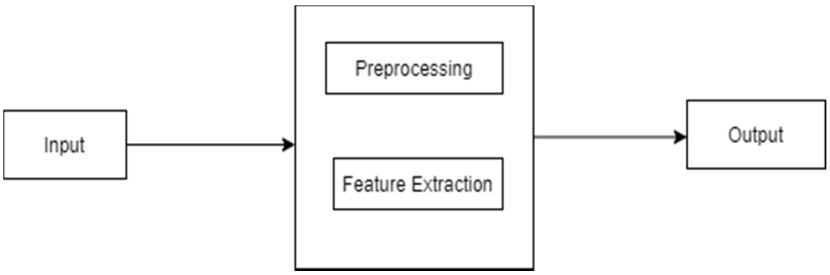


Fig. 3 Data Flow (1) diagram

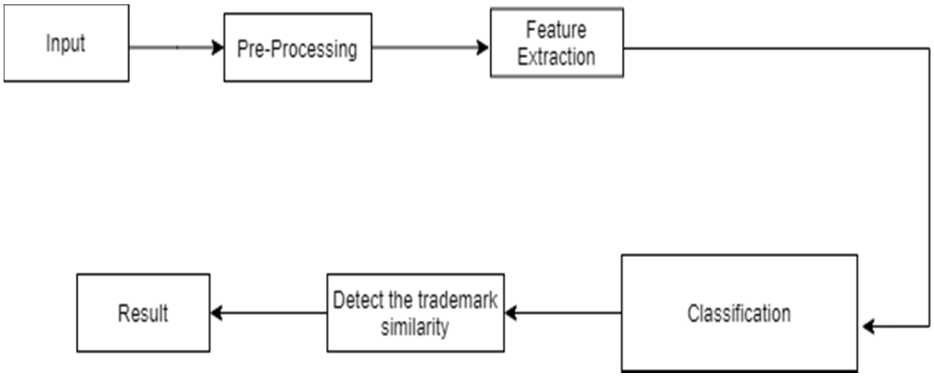


Fig. 4 Data Flow (2) diagram

B. UML Diagrams

Unified Modelling Language is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct and document the artefacts of a software intensive system. UML is process independent, although optimally it should be used in a process that is use case driven, architecture-centric, iterative, and incremental. The Number of UML Diagrams is available.

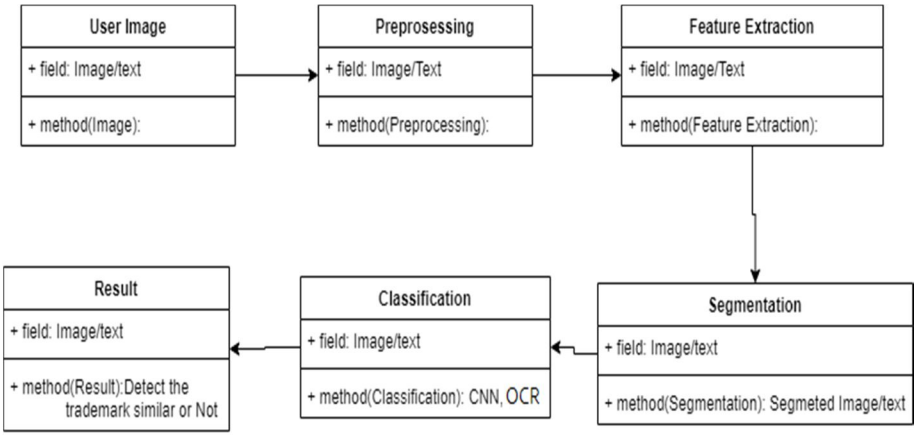


Fig. 5 Class Diagram

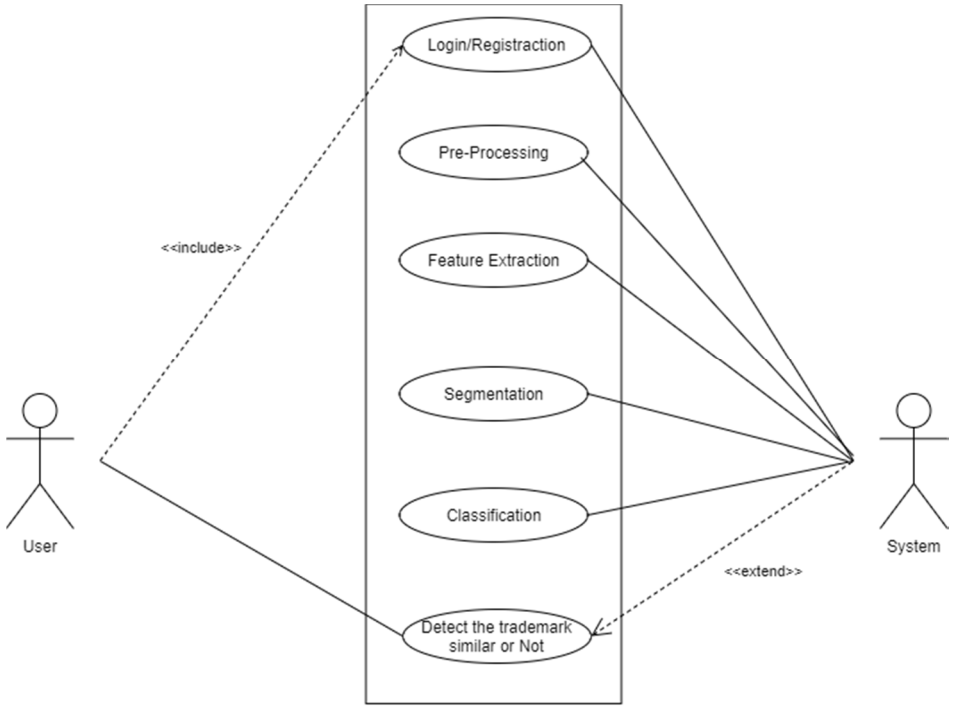


Fig. 6 Use case Diagram

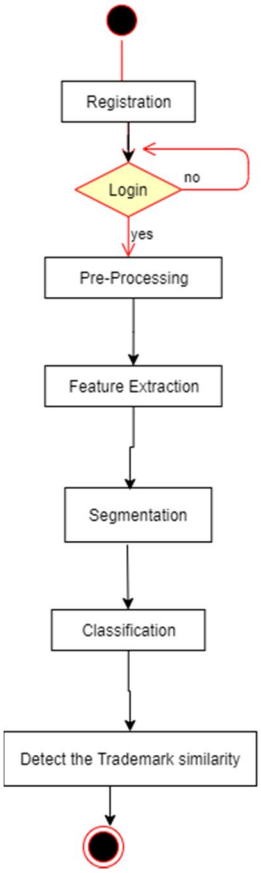


Fig. 7 Activity Diagram

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

V. SOFTWARE REQUIREMENT

A. Python

Python is an interpreter, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was created in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." [30] No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported. Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development. Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his" permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member" Steering Council" to lead the project.

B. Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free. Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages. Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists. When pip installs a package, it automatically installs any dependency Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensor Flow, can find that it stops working having used pip to install a different package that requires a different version of the dependent NumPy library than the one used by Tensor Flow. In some cases, the package may appear to work but produce different results in detail. In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g., the use may wish to have Tensor Flow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done. Open-source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit.

Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of the Python package with conda.

C. Spyder

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. PyCharm is cross-platform, with Windows, MacOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – Features • Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes • Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages • Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others • Support for web frameworks: Django, web2py and Flask [professional edition only] • Integrated Python debugger • Integrated unit testing, with line-by-line code coverage Google App Engine Python development [professional edition only] • Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with change lists and merge. Support for scientific tools like matplotlib, NumPy and SciPy [professional edition only].

VI. RESULT

Fig. 8 Home Page

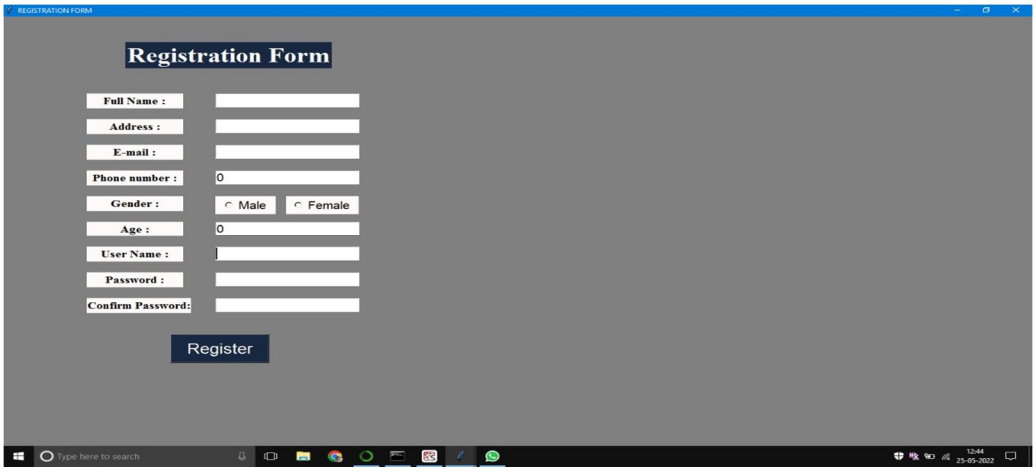


Fig. 9 New User Registration

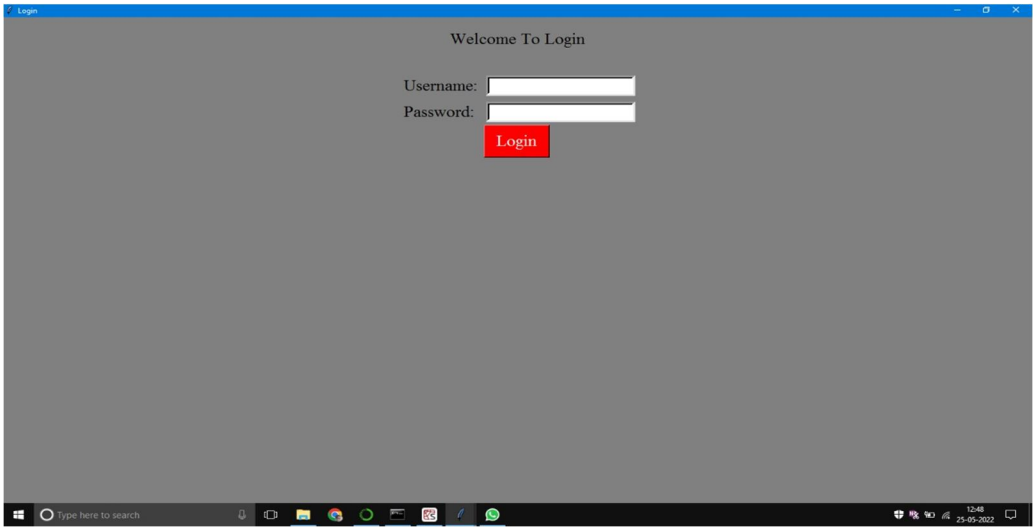


Fig. 10 User Login

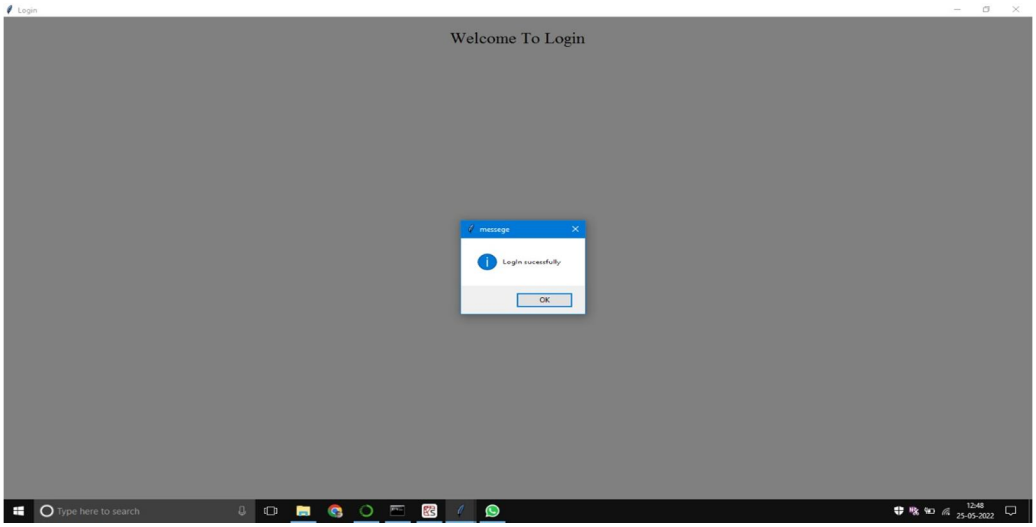


Fig. 11 Login Successfully Message

Fig. 12 Test Trademark Data Input

Fig. 13 Trademark Image Processing

Fig. 14 Trademark Detection Success

Fig. 15 Text Recognition Using OCR in Test Data

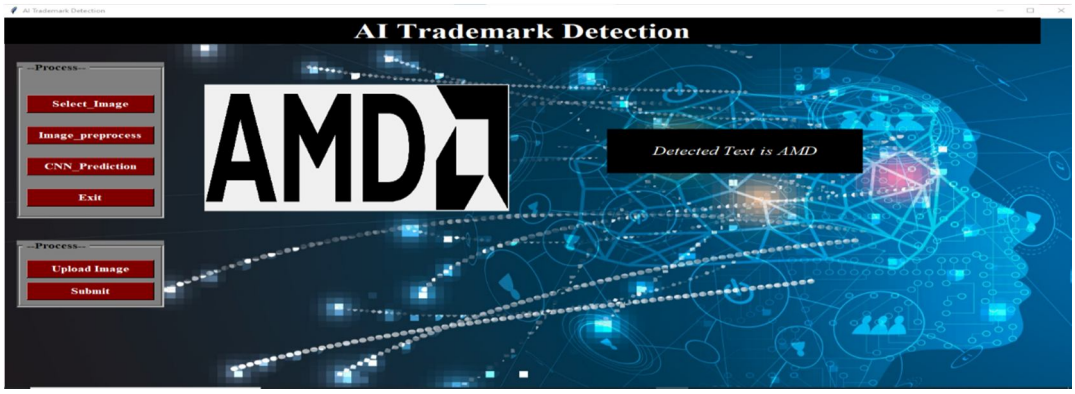


Fig. 16 Text Recognition Using OCR in Test Data

VII. TEST CASE

Test Case ID	Test Case	Test Case I/P	Actual Result	Expected Result	Test case criteria(P/F)
001	Store Xml File	Xml file	Xml file store	Error Should come	P
002	Parse the xml file for conversion	parsing	File get parse	Accept	P
003	Attribute identification	Check individual Attribute	Identify Attributes	Accepted	P
004	Weight Analysis	Check Weight	Analyze Weight of individual Attribute	Accepted	P
005	Tree formation	Form them-Tree	Formation	Accepted	P
006	Cluster Evaluation	Check Evaluation	Should check Cluster	Accepted	P
007	Algorithm Performance	Check Evaluation	Should work Algorithm Properly	Accepted	P
008	Query Formation	Check Query Correction	Should check Query	Accepted	P

Fig. 17 System Test Cases

Test Case ID	Test Case	Test Case I/P	Actual Result	Expected Result	Test case criteria(P/F)
001	Enter the number in username, middle name, last name field	Number	Error Comes	Error Should Comes	P
001	Enter the character in username, middle name, last name field	Character	Accept	Accept	p
002	Enter the invalid email id format in email id field	Kkgmail.com	Error comes	Error Should Comes	P
002	Enter the valid email id format in email id field	kk@gmail.com	Accept	Accept	P
003	Enter the invalid digit no in phone no field	99999	Error comes	Error Should Comes	P
003	Enter the 10 digit no in phone no field	9999999999	Accept	Accept	P

Fig. 18 Registration Test Case

Test Case ID	Test Case	Test Case I/P	Actual Result	Expected Result	Test case criteria(P/F)
001	Enter The Wrong username or password click on submit button	Username or password	Error comes	Error Should come	P
002	Enter the correct username and password click on submit button	Username and password	Accept	Accept	P

Fig. 19 Login Test Case

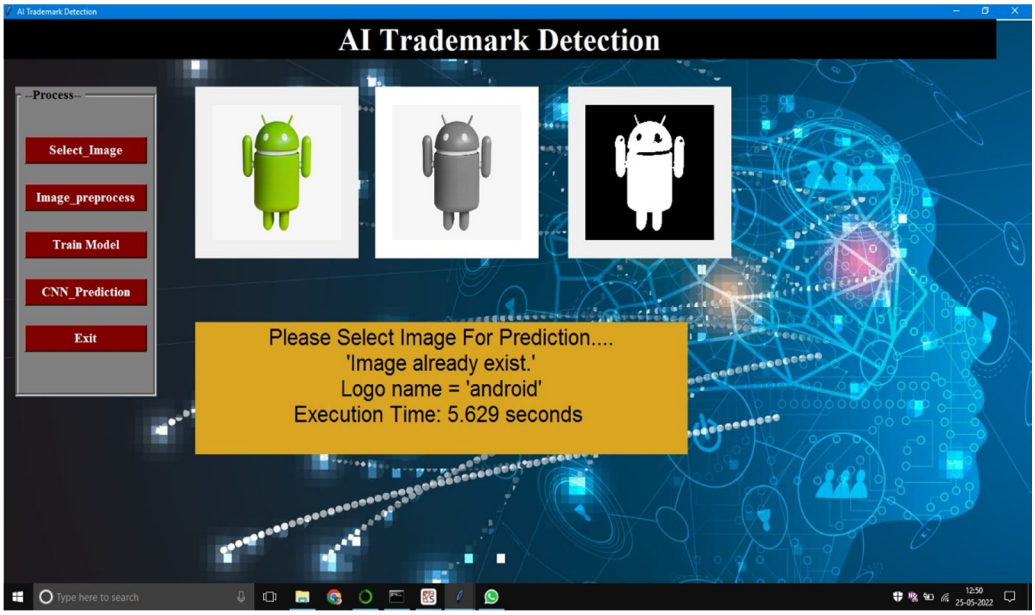


Fig. 20 Trademark Detection Success Message

VIII. CONCLUSION

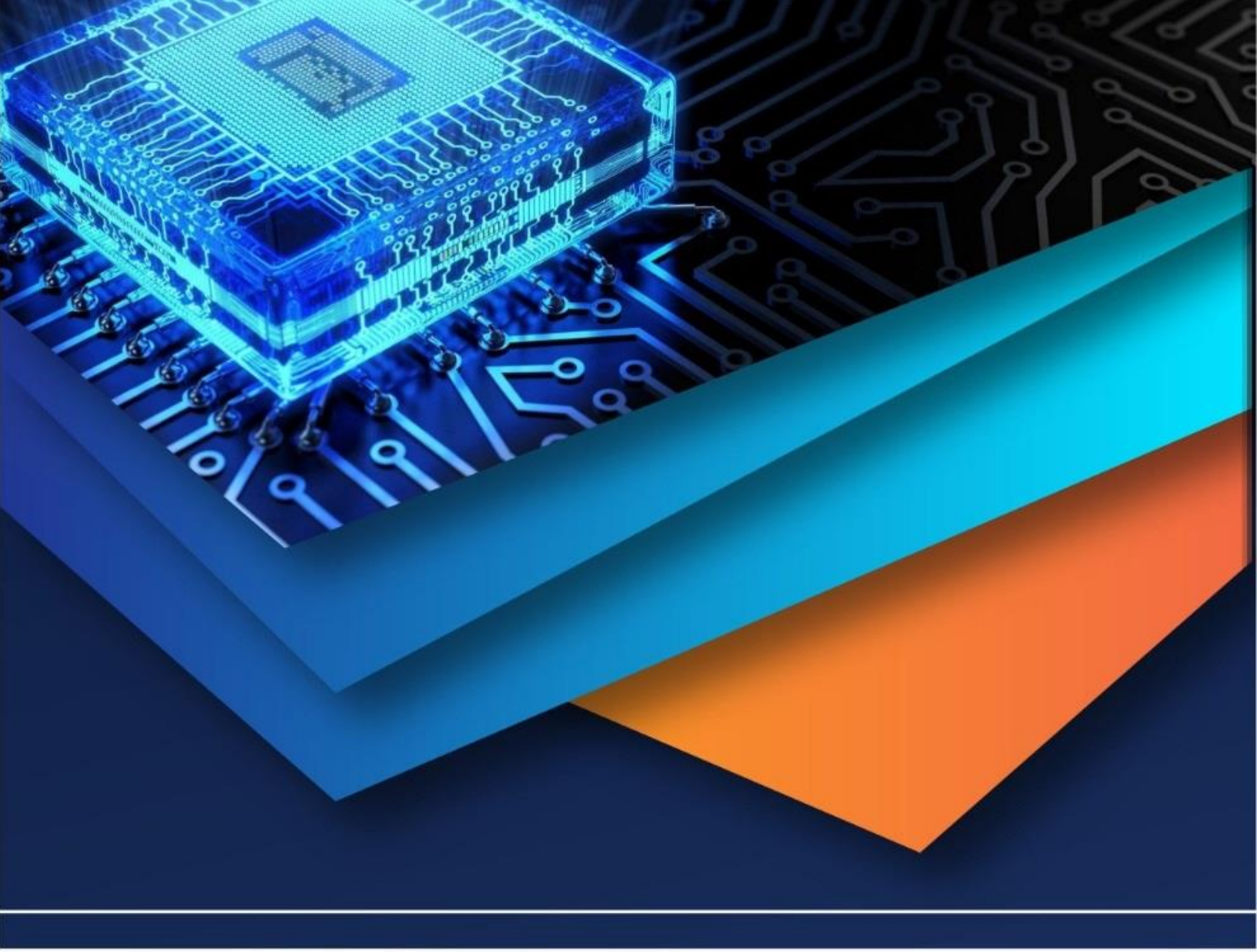
Currently, trademarks examination is a long process that requires manually examining lots of unordered trademarks and the usage of techniques requiring experienced examiners. In this work we introduced several similarity aspects - Automated content similarity, Image/pixel similarity, Text similarity and Manual content similarity - on which we conduct search queries on. Automated and manual content similarities are responsible for catching semantic similarities, while image/pixel similarity is responsible for structural similarities, in contrast to text similarity that seeks for textual similarities and has no correlation to visual similarities. This separation made it possible to fully utilize the advantages of each aspect, as opposed to searching through an unorderly mixture; just through one or an average of them, and thus suffer from a reduction in the significance of similarity between trademarks according to different aspects.

IX. ACKNOWLEDGMENTS

It gives us great pleasure in presenting the preliminary project report on ‘TradeMarker - Artificial Intelligence Based Trademarks Similarity Search Engine’. I would like to take this opportunity to thank my internal guide Prof. Pramod Dhamdhere for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful. I am also grateful to Prof. Shrikant Dhamdhere, Head of Computer Engineering Department, Parvatibai Genba Moze College of Engineering, Wagholi for his indispensable support, suggestions. In the end our special thanks to Other Person Name for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.

REFERENCES

[1] Israeli Trademarks Department, Departments/Trademarks. http://www.justice. gov.il/En/Units/ILPO
[2] Trade Marks Ordinance 5732–1972, s. 8; 11(9); 11(13); 11(14)
[3] Vienna Classification. http://www.wipo.int/classifications/vienna/en
[4] ILPO Annual Report 2016, p. 50 [in Hebrew]
[5] Google VisionAPI. https://cloud.google.com/vision
[6] Lux, M., Chatzichristofis, S.A.: LIRE: Lucene image retrieval: an extensible Java CBIR library. In: Proceedings of the 16th ACM International Conference on Multimedia. ACM (2008)
[7] Lux, M.: Content based image retrieval with LIRe. In: Proceedings of the 19th ACM International Conference on Multimedia. ACM (2011)
[8] Lux, M.: LIRE: open source image retrieval in Java. In: Proceedings of the 21st ACM International Conference on Multimedia. ACM (2013)
[9] Showkatramani, G., Nareddi, S., Doninger, C., Gabel, G., Krishna, A: Trademark image similarity search. In: Stephanidis, C. (ed.) HCI 2018. CCIS, vol. 850, pp. 199–205. Springer, Cham (2018). https://doi.org/10.1007/978- 3-319-92270-6 27
[10] TrademarkVision. https://trademark.vision
[11] NVIDIA article about TrademarkVision. https://news.developer.nvidia.com/ protect-your-trademark-with-artificial-intelligence
[12] IP Australia, Similarity of trade marks. http://manuals.ipaustralia.gov.au/ trademarks/Part 26 Conflict with other Signs - Section 44/5. Similarity of trade marks.html



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY